Apache Tomcat - Servlet and JSP Container

The Apache Tomcat software is an open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies. Tomcat is a Web Container. It also supports JDBC, but does not support EJBs. It could be downloaded from:

http://tomcat.apache.org/

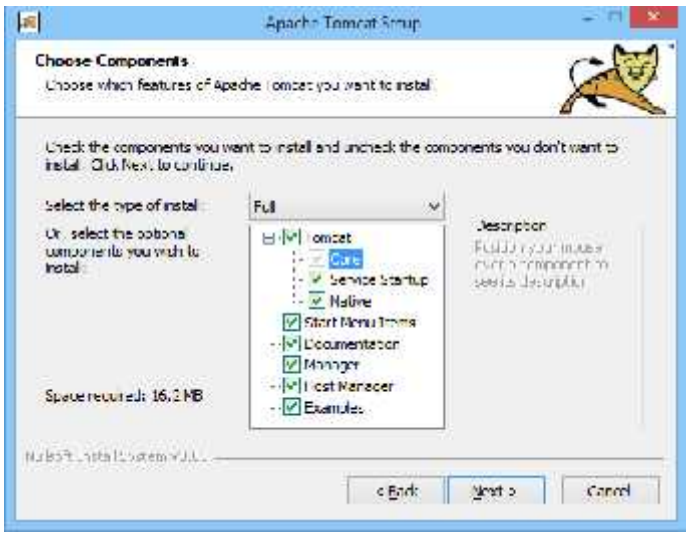
Installation at MS Windows platform

Download the version supporting the required specifications for technologies. That can be checked at page:

http://tomcat.apache.org/whichversion.html

Apache Tomcat 9.x is the current focus of development. It builds on Tomcat 8.0.x and 8.5.x and implements the Servlet 4.0, JSP 2.3, EL 3.0, WebSocket 1.1 and JASPIC 1.1 specifications (the versions required by Java EE 8 platform).

The most simple installing procedure is based on using 32-bit/64-bit Windows Service Installer and starting it from the operating system. During the installation at MS Windows platform it is important to select all components to be installed. **Service Startup** component enables running Tomcat as a service while starting MS Windows (icon will be presented in tray).

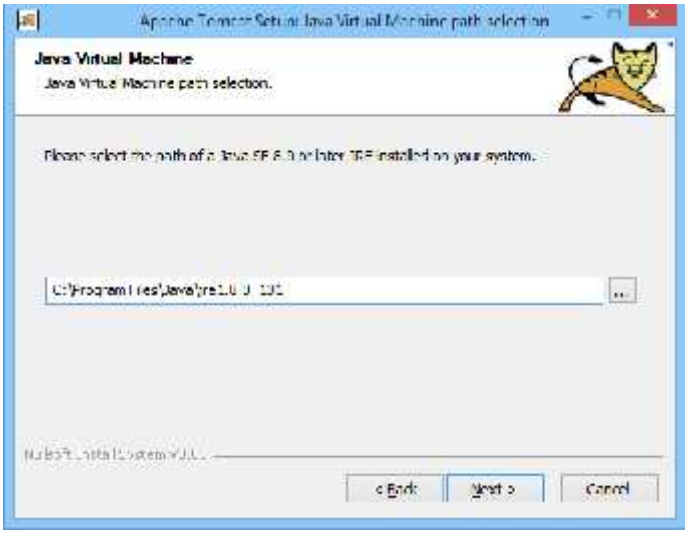


During the setting basic configuration of Tomcat it is important to see that Tomcat will listen HTTP requests at connector port 8080, which means that in all addresses should be included port (:8080).

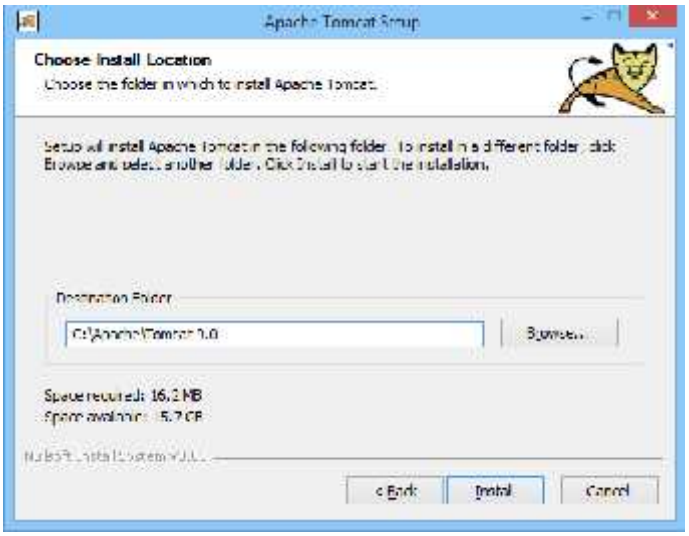
It is important to note that **Tomcat uses 8080 Port for HTTP connections**, which will be used for accessing web applications. This can be checked after completing installation in the file: **$CATALINA_HOME conf\server.xml**

```
<Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
```

The next step is to find and select JRE to be used with Tomcat. The installed version of JRE will be automatically found.



The next step is selection of install folder.



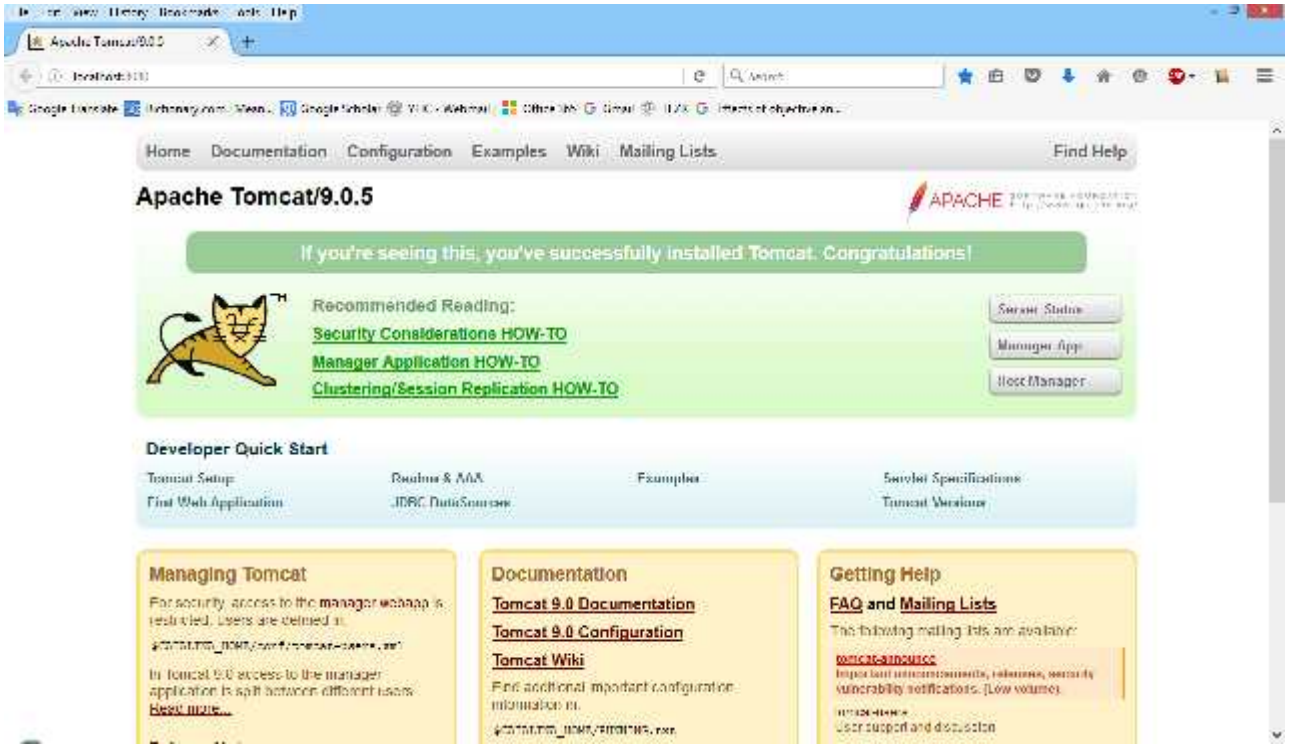
The destination folder should be chosen so that it is easily accessible. For example:

C:\Apache\Tomcat 9.0

After installation it is very useful to read and follow documentation for details about specific issues

https://tomcat.apache.org/tomcat-9.0-doc/index.html

After successful installation Tomcat can be accessed at address: **http://localhost:8080/**



Introduction to Tomcat

Web application installed at Tomcat is called **Context**.

The root of your Tomcat installation is labeled with: **$CATALINA_HOME**. For this installation it is

$CATALINA_HOME = C:\Apache\Tomcat 9.0

Tomcat installed as a service will use a try icon.

The key Tomcat directories:

- **/bin** - Startup, shutdown, and other scripts. The *.sh files (for Unix systems) are functional duplicates of the *.bat files (for Windows systems). Since the Win32 command-line lacks certain functionality, there are some additional files in here.
- **/conf** - Configuration files for managing the server and web applications. The most important file in here is server.xml. It is the main configuration file for the container.
- **/logs** - Log files are here by default.
- **/webapps** - Folder for storing installed web applications. Each web application should be placed in separate folder (context).

Documentation for Web applications at Tomcat

Before starting web development it is important to see information about related technologies:

***JavaServer Pages (JSP) Specification, Version 2.3*.**

http://jcp.org/aboutJava/communityprocess/mrel/jsr245/index2.html

***Servlet API Specification, Version 4.0*.**

https://jcp.org/aboutJava/communityprocess/final/jsr369/index.html

From these locations the specifications in PDF files can be downloaded.

The next important link is to API specifications for Servlets, JSPs and other JEE technologies:

https://javaee.github.io/javaee-spec/javadocs/

To check: Find the package: javax.servlet

Deployment of web application

A web application is defined as a hierarchy of directories and files in a standard layout. Such application hierarchy can be accessed in two forms:

- ***Unpacked form*** - each directory and file exists in the file system separately. This form is usefull during development since each file can be accessed.
- ***Packed form*** - known as a **Web ARchive**, or **WAR** file. This form is useful for distributing applications at web servers.

The top-level directory of your web application hierarchy is also the document root of your application. The top-level directory is used for placing HTML files and JSP pages that comprise your application's user interface. The top-level directory is also a context path to Web application. Thus, if the context path of web application is **/testweb**, then a request URI refers to **/testweb/index.html** and will retrieve **index.html** file from your document root.

Standard directory layout of web application

The directory structure of a web application in unpacked for is:

- ***.html, *.jsp, etc.** - The HTML and JSP pages, along with other files that must be visible to the client browser, such as JavaScript files, CSS files, and images. These files can be placed into a subdirectory hierarchy that reflects web application architecture.
- **/WEB-INF/web.xml - The Web Application Deployment Descriptor** for your application. This is an XML file describing the servlets and other components that make up your application. It contains also initialization parameters and container-managed security constraints.
- **/WEB-INF/classes/** - This directory contains Java class files (and associated resources) required for your application. Here are placed both servlet and non-servlet classes, that are not combined into JAR files. If your classes are organized into Java packages, you must reflect this in the directory hierarchy under /WEB-INF/classes/. For example, a Java class named **`mypackage.MyServlet`** would be stored in a file named **`/WEB-INF/classes/mypackage/MyServlet.class`**.
- **/WEB-INF/lib/** - This directory contains additional JAR files that contain Java class files (and associated resources) required for your application. Here are placed third party class libraries such as JDBC drivers, HTTP upload, XML utilities etc.

After installing an application into Tomcat the classes in the **WEB-INF/classes/** directory and all classes in JAR files found in the **WEB-INF/lib/** directory becomes visible to other classes within web application. This simplifies the installation of web application since there is no need to adjustment of the system class path.

Shared library files

Tomcat also supports mechanisms to install library JAR files (or unpacked classes) once, and make them visible to all installed web applications, without including them inside the web applications. The location for shared code is **$CATALINA_HOME/lib**. JAR files placed here are visible both to web applications and internal Tomcat code. This is a good place to put JDBC drivers that are required for both your application or internal Tomcat use (such as for a JDBCRealm).

Web Application Deployment Descriptor

/WEB-INF/web.xml file contains the **Web Application Deployment Descriptor** for each Web application. This file is an XML document, and defines everything about your application that a server needs to know: servlets, libraries, initialization and security.

The complete syntax and semantics for the deployment descriptor is defined in Servlet API Specification. Example with detailed description is available at link:

https://tomcat.apache.org/tomcat-9.0-doc/appdev/web.xml.txt

The Servlet Specification includes a Document Type Descriptor (DTD) for the web application deployment descriptor, and Tomcat enforces the rules defined here when processing your application's /WEB-INF/web.xml file.

Tomcat Context Descriptor

A **/META-INF/context.xml** file can be used to define Tomcat specific configuration options, such as an access log, data sources, session manager configuration and more. This XML file must contain one Context element, which will be considered as if it was the child of the Host element corresponding to the Host to which the web application is being deployed.

Deployment with Tomcat

A web application must be deployed on a servlet container in order to be executed. During development deployment is used for testing web application. A web application can be deployed in Tomcat by one of the following approaches:

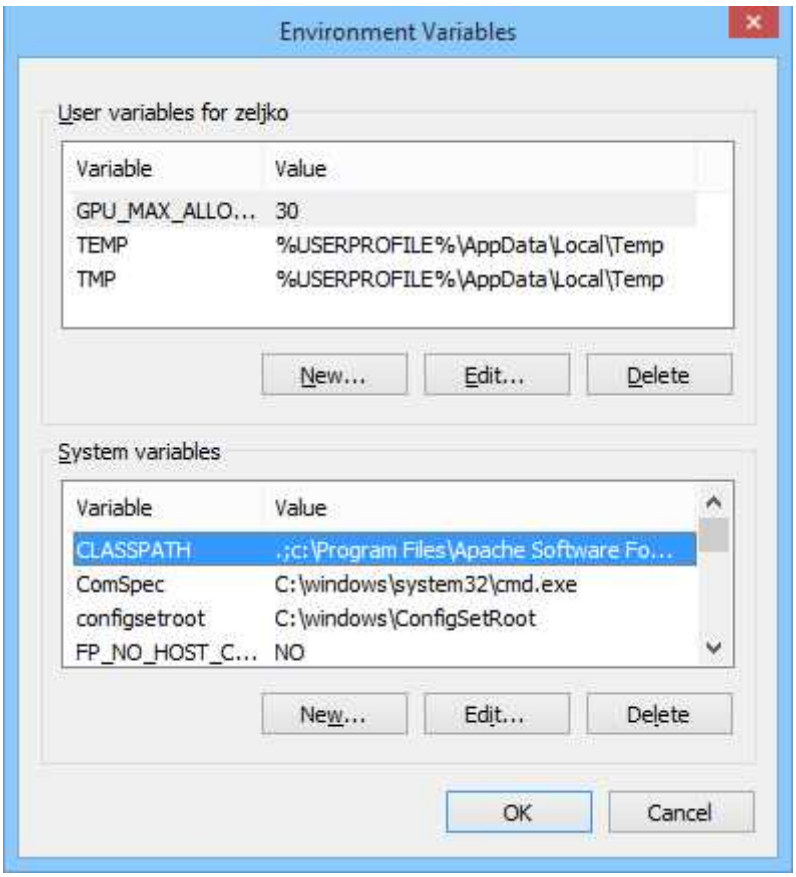
- **Copy unpacked directory hierarchy into a subdirectory in directory $CATALINA_BASE/webapps/**. Tomcat will assign a context path to the application based on the chosen subdirectory name. **After installing or updating the application Tomcat SHOULD BE RESTARTED**.
- **Copy the web application archive file into directory $CATALINA_BASE/webapps/**. When Tomcat is started, it will automatically expand the web application archive file into its unpacked form, and execute the application that way. This approach would typically be used to install an additional application, provided by a third party vendor or by your internal development staff. With this approach you must both replace the web application archive file AND delete the expanded directory that Tomcat created, and then restart Tomcat, in order to reflect your changes.

Organization of web application source

The best development practice suggests separation of the directory hierarchy containing source code from the directory hierarchy containing deployable application. Maintaining this separation has the following advantages:

- The contents of the source directories can be more easily administered, moved, and backed up if the "executable" version of the application is not intermixed.
- Source code control is easier to manage on directories that contain only source files.
- The files that make up an installable distribution of your application are much easier to select when the deployment hierarchy is separate.

Tomcat installs **CLASSPATH** for Servlet and JSP libraries, which can be seen in:



Web development assumes intensive use of API documentation, which is available online:

Apache Tomcat 9.0.5 API

https://tomcat.apache.org/tomcat-9.0-doc/api/index.html

Servlet 4.0 API - Apache Tomcat 9.0.5

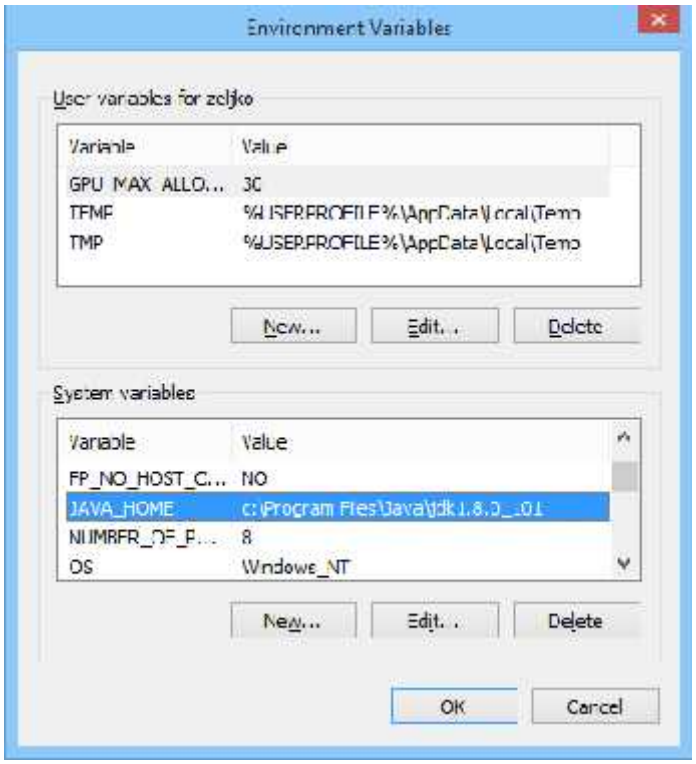
https://tomcat.apache.org/tomcat-9.0-doc/servletapi/index.html

JSP 2.3 API - Apache Tomcat 9.0.5

https://tomcat.apache.org/tomcat-9.0-doc/jspapi/index.html

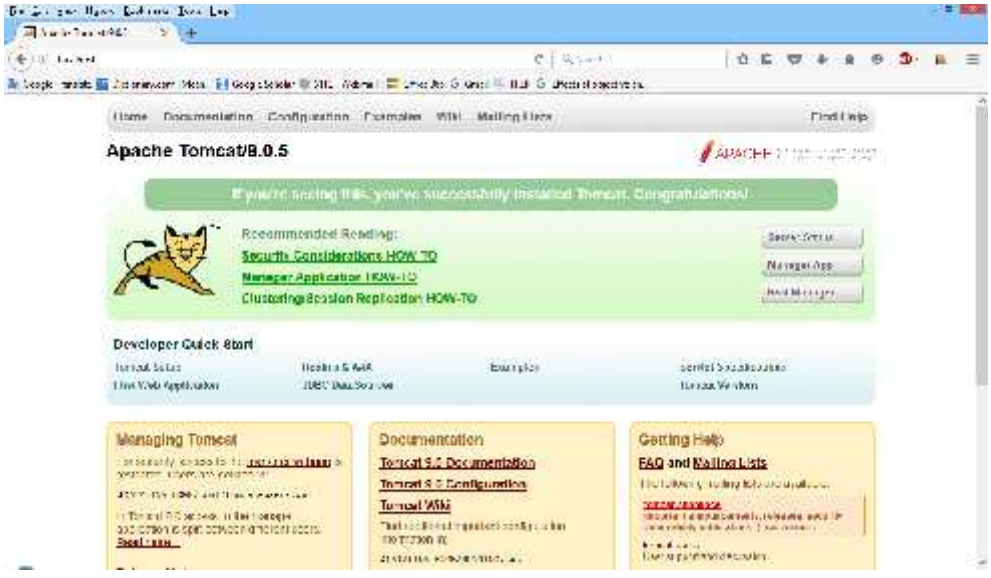
Setting up Tomcat

Setting **JAVA_HOME** variable



Optionally, setting HTTP port to 80 instead 8080 enables calling application without stating port number.

```
    <Connector port="80" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
```



Literature and Links

[1] **The Apache Tomcat**. http://tomcat.apache.org/

[2] **Apache Tomcat 9 Documentation**. https://tomcat.apache.org/tomcat-9.0-doc/index.html

[3] **JavaServer Pages (JSP) Specification, Version 2.3.**
http://jcp.org/aboutJava/communityprocess/mrel/jsr245/index2.html

[4] **Servlet API Specification, Version 4.0**.
https://jcp.org/aboutJava/communityprocess/final/jsr369/index.html

[5] **Java(TM) EE 8 Specification APIs**. https://javaee.github.io/javaee-spec/javadocs/

[6] **Apache Tomcat 9.0.5 API**. https://tomcat.apache.org/tomcat-9.0-doc/api/index.html

[7] **Servlet 4.0 API - Apache Tomcat**. https://tomcat.apache.org/tomcat-9.0-doc/servletapi/index.html

[8] **JSP 2.3 API - Apache Tomcat**. https://tomcat.apache.org/tomcat-9.0-doc/jspapi/index.html